A Small Observatory for Super-Repositories

Lungu, Mircea; Gîrba, Tudor

Published in:
Proceedings of International Workshop on Principles of Software Evolution (IWPSE 2007)

DOI:
10.1145/1294948.1294974

A Small Observatory for Super-Repositories

In Proceedings of International Workshop on Principles of Software Evolution (IWPSE 2007)

Mircea Lungu
Faculty of Informatics
University of Lugano, Switzerland
mircea.lungu@lu.unisi.ch

Tudor Girba
Software Composition Group
University of Bern, Switzerland
girba@iam.unibe.ch

ABSTRACT

Software evolution research has been focused mostly on analyzing the evolution of single software systems. However, it is rarely the case that a project exists as standalone, independent of others. Rather, projects exist in parallel within larger contexts in companies, research groups or even the open-source communities, contexts that we call *super-repositories*. In this paper, we argue that visualization of super-repositories is useful in a range of situations, and we describe The Small Project Observatory , a prototype tool which aims to visualize super-repositories.

1. INTRODUCTION

Researchers mine software project repositories for program comprehension purposes to "learn from history". Visualization plays a key role in this context, because it helps to break down the complexity of the data contained in repositories, leading to a variety of techniques to extract useful information [1, 2, 8, 10, 11].

An underlying assumption of many approaches is that a project represents one single, complex entity that can be analyzed independently. We argue that this is not always the case. Instead, projects exist in larger contexts in companies, research groups or the open-source community and project repositories exist in parallel, in what we call *super-repositories*, that is repositories of project repositories.

Few research projects analyze super-repositories as a whole. One such project is the FlossMole project which provides a database compilation of open-source projects from SourceForge and several other repositories [3]. Weiss analyzed SourceForge only from a pure statistical analysis point of view [12].

In this paper, we argue for the broadening of the perspective and raising the level of abstraction from single project repositories to super-repositories. We claim that such an approach is useful in a variety of contexts: when an open-source contributor is searching for interesting projects to contribute to, when a project manager wishes to supervise multiple projects, or when a new employee wants to understand the "treasure trove" of software that the company has been developing over the years.

For this purpose, we have designed several visualizations and implemented them in a tool called The Small Project Observatory . We provide an overview of our tool and exemplify the visualizations on three open-source super-repositories.

IWPSE'07 September 3-4, 2007, Dubrovnik, Croatia

2. WHY OBSERVE SUPER-REPOSITORIES?

At least three categories of stakeholders can benefit from super-repository visualization: project managers, developers and users. Each of these has different reasons to visualize super-repositories according to his own goals. Several of the possible scenarios are:

Users want to know how serious a project is. In open repositories, we can find a multitude of projects, many of them overlapping in purpose. However, not all of them are equally worthy, and a user needs ways of assurring himself of the seriousness of a project.

Developers want to know what the dependencies are. Not only details of a particular project are relevant, but also the inter-project dependencies are important. For example, in the case of a framework, it is important to know what are the clients so that they can be updated. Similarly, when an application is build out of components, developers need to know what components have changed.

Developers want to know who to ask. One important source of information for developers, are other developers. Thus, developers need to know who to ask [4]. Another possible approach for obtaining information is to ask a question on the mailing-list. However, when getting an answer on the mailing list, the developer wants to know whether the one that answers is knowledgeable in the area.

Project managers want to know how teams work. Projects are developed in teams, and a project manager wants to know how these teams work. While organizational charts can show the team structure on paper, the activity in the repository shows how the actual work gets done [7].

Project managers want to know how projects evolve. Successful projects need to continuously change [9], hence a project manager needs to be up to date with how projects change.

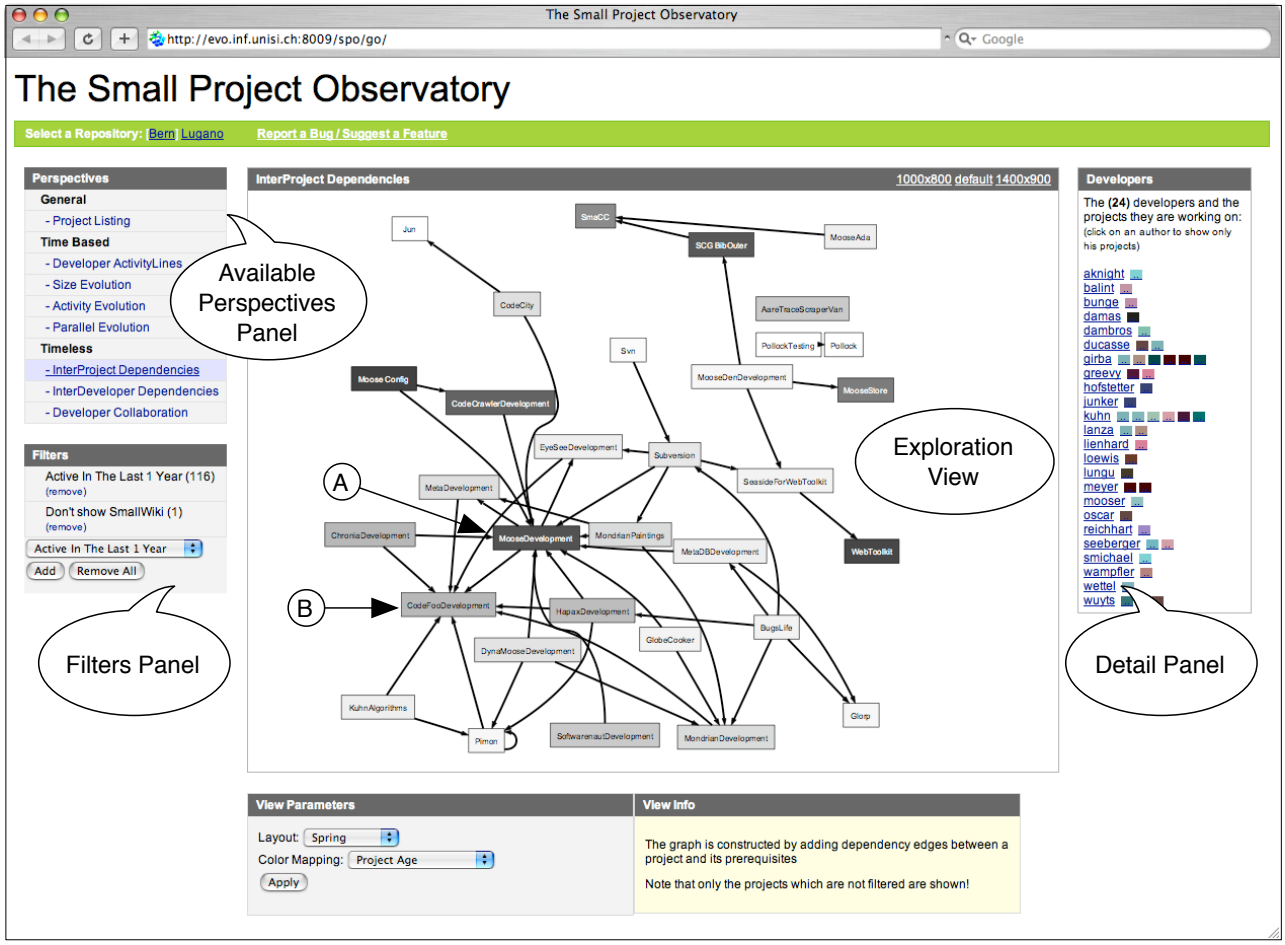

Figure 1: The Web Interface of Small Project Observatory. The figure presents the Inter-Project Dependencies perspective in the Bern super-repository

3. THE SMALL PROJECT OBSERVATORY

Figure 1 presents a screenshot of The Small Project Observatory[1] exploring the Bern super-repository. The Small Project Observatory is a web application that allows us to interactively explore and to filter the super-repository and it offers multiple perspectives.

- **Interaction.** The *Exploration View* presents the currently selected perspective (in this case the dependencies between projects) in a context in which the user can interact with its the elements. The graphical elements can be selected, hoovered over and have contextual menus. The graphics are implemented with SVG[2] and the interaction is generated with Javascript.

- **Predefined Filters.** A variety of predefined filters are part of the application, such as filtering by author, by the status of the project, by the size of the project or by the stage in the release cycle of a project. The user can add and remove predefined filters by using the *Filters Panel*.

- **Multiple Perspectives.** To accomodate the needs of the different stakeholders the tool provides multiple perspectives on the data. The *Available Perspectives Panel* presents the list of available perspectives from which a user can choose.

  To the right of the exploration view there are *Detail Panels* which provide supplementary information on the view or on the selected elements in the view. The detail panel from Figure 1 presents the list of developers which are involved in the projects in the view and the projects they are involved in.

4. VISUAL PERSPECTIVES FOR SUPER-REPOSITORIES

This section presents three of the multiple super-repository perspectives that The Small Project Observatory provides. The perspectives are illustrated with examples from the three Smalltalk project repositories presented in the table below.

[1]Available online at www.inf.unisi.ch/phd/lungu/spo/
[2]Scalable Vectore Graphics (www.w3.org/Graphics/SVG/)

The first is the Open Smalltalk Repository hosted by Cincom[3]. The next two are maintained at the Universities of Bern and Lugano, in Switzerland.

Repository	Projects	Classes	Developers	Since
Cincom	288	19.830	147	2000
Bern	190	10.600	76	2002
Lugano	43	2.088	11	2005

Table 1: The analyzed super-repositories

4.1 Repository Timelines

Purpose: The purpose of the Repository Timelines is to illustrate the evolution of a certain metric at repository level and in the same time, show the contributions of the individual projects to this general evolution. Any metrics which can be aggregated from project-level such as size or activity can visualized this way.

Construction: The total time interval of interest is divided in smaller time units such as months, weeks, or days depending on the granularity of the desired analysis. Each project is assigned a specific color and it is represented as a surface where the horizontal axis shows time and the height of the surface is given at every point in time by the value of the metric for the project at that point in time. The projects are stacked one on top of the other in cronological order with the oldest ones at the bottom. The perspective also emphasizes the specific time intervals when each project's size changes: the brightness of the project color is higher in the periods when the size remains constant.

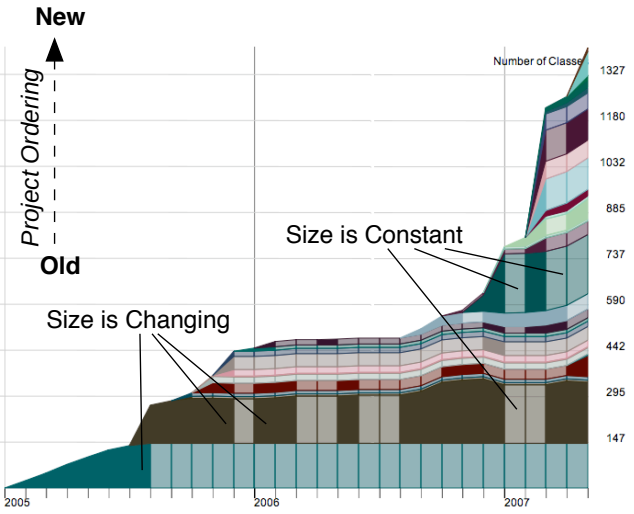

Figure 2: Size Timeline in Lugano

Example: Figure 2 illustrates the the Project Size Timeline for the projects in the Lugano super-repository. Since we are dealing with a repository of projects written in an object-oriented languages, we consider Number of Classes to be a good estimation [6]. One can notice the fact that the repository is growing with different speeds over time. After about one year of slow growth the size jumps dramatically at the begining of 2007.

Another interesting observation is that in this repository there are several projects which grow for a short period of time and freeze after that. To find out more about these projects the user can zoom in and focus on them.

4.2 Inter-Project Dependency Map

Purpose: The Inter-Project Dependency Map presents the static dependencies between the projects in a super-repository. Depending on the type of repository and on the languages in which the projects are developed the dependency information can be computed in various ways. In the case of Store repositories, every project can explicitly list the projects that it depends on.

Construction: The visual representation is a graph where projects are nodes and dependencies between them are arows. Various metrics, computed for the individual projects, are mapped on the color of the projects. Such an overview pinpoints the critical projects in a company (*i.e.,* projects that cannot die) or can be useful in assesing the relevance of the project. Indeed even if there is low activity in a project, if there are many projects depending on it, it is a safe bet that the project is relevant.

Example: Figure 1 shows a screenshot of our tool displaying a dependency map for all the projects from the Bern repository which were active in the last year. The color of the project is proportional to its age: the older the project, the darker the color. The figure shows that the most depended on project 'MooseDevelopment' (marker A) which a Timeline View shows to be the oldest active project in the repository.

4.3 Developer Collaboration Map

Purpose: The purpose of the Developer Collaboration Map is to provide a visualization of the inter-developer collaborations as they can be inferred from the history of all the projects in the super-repository.

Construction: We say that two developers *collaborate* on a certain project if, during the lifetime of that project, they both make modification to the project for a certain number of times which is greater than a given threshold. We call this metric the *developer commit count (DCC)*. Based on this information we can construct a *collaboration graph* where the nodes are developers and the edges between them represent projects on which they collaborate.

To represent the collaboration graph for a super-repository we draw the graph using a *force-based layout algorithm* which clusters connected nodes together and offers an aesthetically pleasing layout [5]. Thus, developers which collaborate will be positioned closer together. Complementary to the positioning information, the color of an edge emphasizes the project which determines that collaboration. We call this a *collaboration map* of the repository.

Example: Figure 3 presents the collaboration maps of two of the analyzed repositories: the left is the Cincom repository and the right is the Bern repository. In both the cases we have considered a contributor to be a developer if he had $DCC > 15$. One surprising observation is that the Bern super-repository, although smaller, contains more collaborative projects than the Cincom. Another observation is that in the case of Bern there are a number of developers which are very central to the network as they collaborate

[3]http://smalltalk.cincom.com

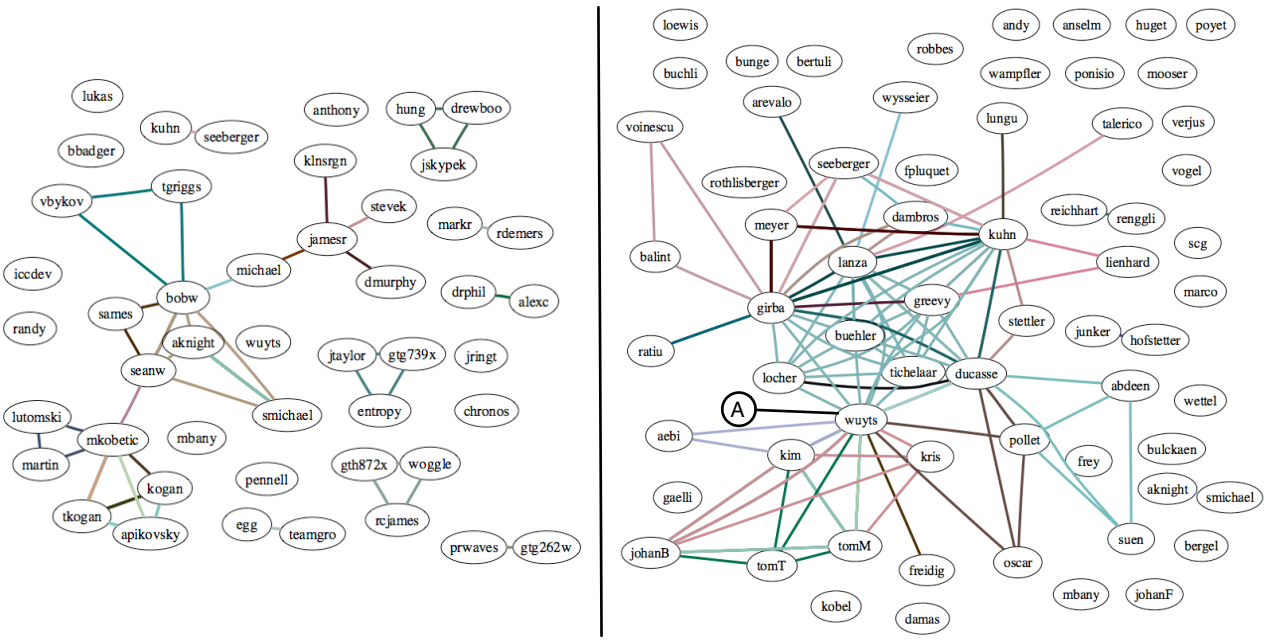

Figure 3: Collaboration between the developers in the Cincom and Bern super-repositories

with many other on many projects (*e.g.,* marker A).

5. CONCLUSIONS

In this article we have argued that interactive visualization is important for understanding super-repositories and we have presented The Small Project Observatory a prototype tool which supports such visualizations.

Acknowledgments We would like to thank Jochen Wuttke and Daniel Ratiu for discussions on earlier versions of this article. We also gratefully acknowledge the financial support of the Swiss National Science Foundation for the project "NOREX — Network of Reengineering Expertise" (SNF Project IB7320-110997).

6. REFERENCES

[1] T. Ball and S. Eick. Software visualization in the large. *IEEE Computer*, 29(4):33–43, 1996.

[2] C. Collberg, S. Kobourov, J. Nagra, J. Pitts, and K. Wampler. A system for graph-based visualization of the evolution of software. In *Proceedings of the 2003 ACM Symposium on Software Visualization*, pages 77–86, New York NY, 2003. ACM Press.

[3] M. Conklin, J. Howison, and K. Crowston. Collaboration using ossmole: a repository of floss data and analyses. *SIGSOFT Softw. Eng. Notes*, 30(4):1–5, 2005.

[4] D. Cubranic and G. Murphy. Hipikat: Recommending pertinent software development artifacts. In *Proceedings 25th International Conference on Software Engineering (ICSE 2003)*, pages 408–418, New York NY, 2003. ACM Press.

[5] T. M. J. Fruchterman and E. M. Reingold. Graph drawing by force-directed placement. *Softw. Pract. Exper.*, 1991.

[6] T. Gîrba, S. Ducasse, and M. Lanza. Yesterday's Weather: Guiding early reverse engineering efforts by summarizing the evolution of changes. In *Proceedings of 20th IEEE International Conference on Software Maintenance (ICSM'04)*, pages 40–49, Los Alamitos CA, Sept. 2004. IEEE Computer Society.

[7] T. Gîrba, A. Kuhn, M. Seeberger, and S. Ducasse. How developers drive software evolution. In *Proceedings of International Workshop on Principles of Software Evolution (IWPSE 2005)*, pages 113–122. IEEE Computer Society Press, 2005.

[8] M. Jazayeri, H. Gall, and C. Riva. Visualizing Software Release Histories: The Use of Color and Third Dimension. In *Proceedings of ICSM '99 (International Conference on Software Maintenance)*, pages 99–108. IEEE Computer Society Press, 1999.

[9] M. Lehman and L. Belady. *Program Evolution: Processes of Software Change*. London Academic Press, London, 1985.

[10] M. Pinzger, H. Gall, M. Fischer, and M. Lanza. Visualizing multiple evolution metrics. In *Proceedings of SoftVis 2005 (2nd ACM Symposium on Software Visualization)*, pages 67–75, St. Louis, Missouri, USA, May 2005.

[11] F. Van Rysselberghe and S. Demeyer. Studying software evolution information by visualizing the change history. In *Proceedings 20th IEEE International Conference on Software Maintenance (ICSM '04)*, pages 328–337, Los Alamitos CA, Sept. 2004. IEEE Computer Society Press.

[12] D. A. Weiss. A large crawl and quantitative analysis of open source projects hosted on sourceforge. In *Report 001/05, Pozna University of Technology, Poland*, 2005.